REMARKS

This is intended as a full and complete response to the Office Action dated July 21, 2008, having a shortened statutory period for response set to expire on October 21, 2008. Please reconsider the claims pending in the application for reasons discussed below.

Claims 1-3, 5, 6, 9-12, 14-18, 20, 21 and 24-27 are pending in the application. Claims 1-3, 5, 6, 9-12, 14-18, 20, 21 and 24-27 remain pending following entry of this response. Claim 12, 16, 18, 20, 21 & 24-27 have been amended. Applicants submit that the amendments and new claims do not introduce new matter.

Claim Rejections - 35 U.S.C. § 103

Claims 1-3, 5, 9, 12, 14, 16-18, 20, and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Applicant Admitted Prior Art in Applicant's Specification (paragraphs 8, 9, and 35) ("AAPA") in view of *Veditz et al.* ("*Veditz*"), US Patent No. 6,496,793. Applicants respectfully traverse this rejection.

The Examiner bears the initial burden of establishing a prima facie case of obviousness. See MPEP § 2141. Establishing a prima facie case of obviousness begins with first resolving the factual inquiries of Graham v. John Deere Co. 383 U.S. 1 (1966). The factual inquiries are as follows:

(A) determining the scope and content of the prior art;
(B) ascertaining the differences between the claimed invention and the prior art;
(C) resolving the level of ordinary skill in the art; and
(D) considering any objective indicia of nonobviousness.

Once the Graham factual inquiries are resolved, the Examiner must determine whether the claimed invention would have been obvious to one of ordinary skill in the art.

Respectfully, Applicants submit that the Examiner has not properly characterized the teachings of the references and/or the claims at issue. Accordingly, a prima facie case of obviousness has not been established.

In this particular case, the Examiner attempts to leverage a BPAI decision affirming a rejection of the claims as presented prior to the amendments submitted in an RCE following that Board decision. As demonstrated below, however, AAPA, in view of *Veditz* does not render the present claims obvious. In particular, AAPA does not disclose a "method of determining an appropriate character set for use in client-server communications," that includes:

> (a) selecting a character set for a client request made by client to a server using a network communication protocol, the selecting comprising:
>
> determining whether the client request includes, as part of the network communication protocol, a request character set designation; and
>
> if the client request does not include the request character set designation:
>
> (i) retrieving locale information contained in the client request;
>
> (ii) selecting a character set to assign to the request character set designation by associating the locale information with the request character set designation using mapping data located on the server; and
>
> (iii) associating the request character set designation with a first code-set converter designation, wherein the first code-set converter designation is contained in a lookup table and is mapped in the lookup table with the character set assigned to the request character set designation, and wherein a first code-set converter corresponding to the first code-set converter designation maps characters of the request character set designation to corresponding characters of the first code-set converter designation while processing the request.

Claims 12 and 16 recite similar limitations. In rejecting these limitations of claims 1, 12, and 16, the Examiner relies on statements in paragraphs [0008], [0009], and [0035]. Specifically, the Examiner suggests as follows regarding claim 12.

> Regarding claim 12, AAPA teaches a server computer system connected to at least one client computer, the server computer system comprising a memory containing a code-set program and at least one processor, wherein the processor, when executing the code-set program (see Specification, paragraphs 8, 9, and 35).
> is configured to:
>
> …

> if the request header does not designate the character set:
> (i) *retrieve locale information from the client request* (See Specification, paragraphs 8,9, and 35. Specifically in paragraph 35 Appellant admits that determining that a client request does not designate a character set, a well known API, developed by Sun Microsystems may be invoked to retrieve locale information from the client request in order to determine an associated character set).
> (ii) *associate the locale information with a character set* (See Specification, paragraphs 8,9, and 35. Specifically in paragraph 35 Appellant admits that determining that a client request does not designate a character set, a well known API, developed by Sun Microsystems may be invoked to retrieve locale information from the client request in order to determine an associated character set).

Office Action, p. 4. Regarding claims 1 and 16, the Examiner suggests "Independent claims 1 and 16 incorporate substantially similar subject matter as independent claim 12, and are rejected along the same rationale." Office Action, p. 5. However, claim 1 does not recite "associating the locale information with a character set," as claim 12 does. Instead, claim 1 was amended in an RCE to recite "selecting a character set to assign to the request character set designation by associating the locale information with the request character set designation using mapping data located on the server[1]."

Further, Applicants submit that paragraph [0035] does not teach that "a well known API, developed by Sun Microsystems may be invoked to retrieve locale information from the client request in order to determine an associated character set." Set out in full, paragraph [0035] provides:

> **[0035]** If the "Content-Type" header is missing from an HTTP request, or if the "Content-type" header does not contain a code-set identifier, the computer program 110 determines the locale of the HTTP request by invoking an Application Programming Interface (API) 129 configured to extract the locale from the HTTP request. One API which may be used to advantage is the ServletRequest.getLocale() API developed by Sun Microsystems. If the Accept-Language HTTP input header contains the most preferred cultural setting of the client, the API 129 returns that cultural preference. Otherwise, it returns the server's locale as the default. The computer program 110 selects the appropriate character set associated with the locale identifier returned by the API 129.

---

[1] Additionally, with this response, the corresponding limitation of claim 12 has been amended to recite the same language as claim 1.

Specification, ¶ 35. However, as pointed out in paragraph [0035], the "ServletRequest.getLocale()" API call returns locale information not character set information. Plainly, the "ServletRequest.getLocale()" API cannot be "invoked … in order to determine an associated character set," as suggested. For example, documentation published by Sun Microsystems[2] describes the functionality of this ServeletRequest.getLocale() API () provides as follows:

> **getLocales**
> `public java.util.Enumeration` **`getLocales`**`()`
>
> Returns an `Enumeration` of `Locale` objects indicating, in decreasing order starting with the preferred locale, the locales that are acceptable to the client based on the Accept-Language header. If the client request doesn't provide an Accept-Language header, this method returns an `Enumeration` containing one `Locale`, the default locale for the server.
>
> **Returns:**
>
> an `Enumeration` of preferred `Locale` objects for the client

As the Sun documentation makes clear, the getLocales() call returns an enumeration of "Locale" objects. Thus, while "ServletRequest.getLocale()"may be invoked to obtain "locale information" as suggested, nothing about this API call discloses anything whatsoever to do with character sets. Consider an analogy: assume that a request sent to a remote system included a person's address and phone number, but lacked an area code. Assume further that the "getLocales()" call returned a person's zip code contained in the request. In such a case, the "getLocales()" call, while functioning as intended, would not by itself provide a means to call back a person who submitted the request, as invoking the "getLocales()" call returns zip codes, not area codes (at least in this example). In reality, of course, the "getLocales()" call returns locales, not character sets. Nevertheless, like the analogy of a zip code and an area code, nothing about a value for a "locale" returned from an invocation of "getLocales()" would be "associated with a character set," as suggested.

Claim 1 (as well as claims 12 and 16) recite "(ii) selecting a character set to assign to the request character set designation by associating the locale information

---

[2] See http://java.sun.com/j2ee/1.4/docs/api/javax/servlet/ServletRequest.html#getLocales()

with the request character set designation using mapping data located on the server." While the Examiner is correct that the "ServeletRequest.getLocale()" could be used to perform the claimed step of retrieving locale information contained in the client request (in fact, this is expressly contemplated by Application, paragraph 35), nothing in Applicants description of the ServeletRequest.getLocale()" (or the actual documentation from Sun) demonstrates that the claimed step of "selecting a character set to assign to the request character set designation by associating the locale information with the request character set designation using mapping data located on the server" was known.

> Further, claim 1 goes on to recite:
>
> (iii) associating the request character set designation with a first code-set converter designation, wherein the first code-set converter designation is contained in a lookup table and is mapped in the lookup table with the character set assigned to the request character set designation, and wherein a first code-set converter corresponding to the first code-set converter designation maps characters of the request character set designation to corresponding characters of the first code-set converter designation while processing the request.

Claims 12 and 16 recite a similar limitation. The Examiner does not suggest that Applicants "admitted prior art" discloses this limitation; however, the Examiner suggests:

> [*Veditz*] (iii) associating the character set with a code-set converter designation, ... (See fig. 28 and 2C - i.e. "LDID Value"; see also col. 13, lines 1-67 to col. 14, lines 1-62 where each character set is associated with a code-set designation in a lookup table that maps the associations).

Office Action, p. 4. However, *Veditz*, columns 13 and 14 provides a header file of definitions specifying a relationship between what *Veditz* describes as "language driver" and a "code page." In other words, the mappings specify "for language driver X use code page Y". As used in *Veditz*:

> "Language drivers" are provided to correctly handle characteristics of a given language. The drivers reference a character set and a collection of tables describing the rules for that character set. For instance, language drivers include information about character sets (code pages), sorting orders, upper case and lower case rules, which characters are alphabetic, and what double-letter combination it is to accept.

Veditz, 11:59-65. And *Veditz* describes a "Language Driver Identifier" or "LDID" as an element used to identify a given "Language Driver;" specifically:

> Particularly for those embodiments having data objects constrained by downward compatibility or storage space considerations, the descriptor is a Language Driver Identifier (LDID) of the present invention. The LDID may be embodied in the form of a system-comparable unit, such as an ID byte which references an agreed-upon set of values (e.g., locale lookup table). For purposes of clarity, the discussion which follows will focus on use of the LDID descriptor embodied as a byte identifier.

Veditz, 12:43-53. Thus, the "LDID" value cited to be the examiner – merely provides a byte value used to identify a particular "Language Driver." At the same time, nothing about the table in *Veditz*, col. 12-13 which lists a "one-to-one correspondence between a language driver and its LDID," discloses anything about mapping from one character set to another character set in general, and does not disclose the claimed limitation of "a first code-set converter corresponding to the first code-set converter designation" which "maps characters of the request character set designation to corresponding characters of the first code-set converter designation while processing the request." Instead, the LDID value may be used to indentify a "language Driver, and as stated the language driver "reference a character set and a collection of tables describing the rules for that character set. For instance, language drivers include information about character sets (code pages), sorting orders, upper case and lower case rules, which characters are alphabetic, and what double-letter combination it is to accept." *Veditz*, 11:59-65.

Accordingly, for all the foregoing reasons, Applicants submit that Applicant "admitted prior art" in view of *Veditz*, does not disclose the limitations of claims 1, 12, or 16. Therefore, these claims (and claims dependent therefrom) are believed to be allowable, and allowance of the claims is respectfully requested.

Claims 6, 10, 11, 21, 26, and 27 are rejected under 35 U.S.C. 103(a) as being unpatentable over Applicant Admitted Prior Art in Applicant's Specification (paragraphs 8, 9, and 35) ("AAPA") in view of *Veditz et al.* ("*Veditz*"), US Patent No. 6,496,793 and further in view of *Horn et al.*, ("*Horn*"), US PGPUB No. 2002/0156688.

Claims 6, 10, 11, 21, 26, and 27 each depend from one of independent claims 1 or 16. As the discussion above demonstrates that Applicant "admitted prior art" in view of *Veditz*, does not disclose the limitations of these independent claims, Applicants submit that dependent claims 6, 10, 11, 21, 26, and 27 are allowable without the need for further comment.

Claims 15 and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Applicant Admitted Prior Art in Applicant's Specification (paragraphs 8, 9, and 35) ("AAPA") in view of *Veditz et al.* ("*Veditz*"), US Patent No. 6,496,793, and further in view of *Kan et al.*, ("*Kan*"), US PGPUB No. 2003/0088544.

Claims 6, 10, 11, 21, 26, and 27 each depend from one of independent claims 1 or 16. As the discussion above demonstrates that Applicants "admitted prior art" in view of *Veditz*, does not disclose the limitations of these independent claims, Applicants submit that dependent claims 6, 10, 11, 21, 26, and 27 are allowable without the need for further comment.

Claims 1, 3, 5, 9, 12, 14, 16, 18, 20, and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Veditz et al.* ("*Veditz*"), US Patent No. 6,496,793, in view of *Watanabe et al.* ("*Watanabe*"), US Patent No. 6,185,729.

Claims 2, 6, 10, 11, 17, 21, 26, and 27 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Veditz et al.* ("*Veditz*"), US Patent No. 6,496,793, in view of *Watanabe et al.* ("*Watanabe*"), US Patent No. 6,185,729, and further in view of *Horn et al.*, ("*Horn*"), US PGPUB No. 2002/0156688.

As noted above, claims 1, 12, and 16 each recite a step of

> (iii) associating the request character set designation with a first code-set converter designation, wherein the first code-set converter designation is contained in a lookup table and is mapped in the lookup table with the character set assigned to the request character set designation, and wherein a first code-set converter corresponding to the first code-set converter designation maps characters of the request character set designation to corresponding characters of the first code-set converter designation while processing the request.

In rejecting claims 1, 12 and 16, the Examiner suggests:

> [*Veditz*] (iii) associating the character set with a code-set converter designation, ... (See fig. 28 and 2C - i.e. "LDID Value"; see also col. 13, lines 1-67 to col. 14, lines 1-62 where each character set is associated with a code-set designation in a lookup table that maps the associations).

Office Action, p. 11. For all the reasons set forth above regarding these same passages, the LDID value provides an identifier for a "language driver"; nothing about the table in *Veditz*, col. 12-13 which lists a "one-to-one correspondence between a language driver and its LDID," discloses anything about mapping from one character set to another character set in general, and does not disclose the claimed limitation of "a first code-set converter corresponding to the first code-set converter designation" which "maps characters of the request character set designation to corresponding characters of the first code-set converter designation while processing the request." Instead, the LDID value may be used to indentify a "language Driver, and as stated the language driver "reference a character set and a collection of tables describing the rules for that character set. For instance, language drivers include information about character sets (code pages), sorting orders, upper case and lower case rules, which characters are alphabetic, and what double-letter combination it is to accept." *Veditz*, 11:59-65.

Further, Applicants submit that *Veditz* does not teach the claimed step of "determining whether a client request includes ... a request character set designation, and if the client request does not include the request character set designation, (i) retrieving locale information contained in the client request and (ii), selecting a character set to assign to the request character set designation by associating the locale information with the request character set designation using mapping data located on the server.

The Examiner suggests:

> [*Veditz* discloses] selecting a character set to assign to the request character set designation by associating the locale information with the request character set designation using mapping data located on the server (Fig. 2B - if Active LDID is not equal to Local LDID it maps the Local LDID into the Active LDID; see also col. 3, lines 54-60; col. 7, lines 5264; col. 18, lines 21-26); and

Office Action, p. 11. Generally, *Veditz* discloses a non-networked system that will "intelligently process data objects created or modified under one language driver with those created or modified by a different language driver." *Veditz*, Abstract. The system disclosed by *Veditz* "continually checks and maintains correct language configuration. The LDID may be used to identify a "langue driver that was in use when the data object was created or modified." *Veditz*, 3:23-28.

This process of maintaining a correct language configuration enables the system of *Veditz* to determine when the system is inappropriately configured for a data object about to be processed. *Veditz*, 7:45-50. Each data object in the system may be tagged with an LDID, and the system itself maintains an "active" LDID (i.e., the LDID presently being used by the system). *Veditz*, 14:56-62. The active LDID, in turn, is written to data objects which the system "touches" (i.e., creates or modifies). *Veditz,* 14:63-64. In the event of a mismatch between the "active" LDID and the LDID of a "data object," corrective actions may be taken.

The system disclosed in *Veditz* requires data objects be "tagged" with an LDID value in order to function. The present claims, however, are directed to determining a character set when a "client request" *expressly fails* to include a character set designation. For example, claims 1 and 16 recite: "determining whether the client request includes, as part of the network communication protocol, a request character set designation; and if the client request *does not* include the request character set designation [performing subsequent steps]."

One of the subsequent steps recited by claims 1 and 16 recite: "retrieving locale information contained in the client request …." In other words, when a character set designation is not included in the request, retrieve something else *from the client request*; in this case "locale information" contained in the client request. On this point, the Examiner cites *Veditz* Figure 3B and states "Fig. 3B → compares LDID of data file to Active LDID; see also col. 3, lines 29-31" *See Final Office Action*, p. 3. The passage cited by the Examiner, however, merely describes that the LDID may be represented using a byte (i.e., 8 bits of data):

> The LDID, which may be in the form of an ID byte, references a set of language driver values (e.g., lookup table of locales).

Further, Figure 3 is a flow chart "illustrating a language-dependent file operation method of the present invention." *Veditz*, 4:20-21. The method generally includes comparing a data value from a database header file with the current "active LDID" maintained by the system. If no "LDID value" is included in the data object, *Veditiz* discloses that a database object may be opened using a read-only access mode. *See, e.g., Veditz*, Figure 3A, 304.

If an "LDID value" is stored with the data object, *Veditz* discloses comparing this value with the "Active LDID. *See e.g., Veditz,* Figure 3B. The "Active LDID" value, however, is not "*retrieved from the client request*" as recited by claims 1, 12, and 16; instead the "Active LDID" is a value maintained by the system for "tagging" each "data object" modified during a given session. For example, *Veditz*, Figure 2A illustrates the "active LDID" element as part of the language configurator 230. Separately, Figure 2B also illustrates a data object 201, with a local LDID 215 in a header file. Even assuming that the "data object" illustrated in Figure 2 reads on the recited "client request," clearly the "active LDID" is not retrieved *from the data object* 201. Quite the contrary, the "active LDID" is a system variable independent of any particular data object or request. Thus, Applicants respectfully submit that the material cited by the Examiner fails to disclose recited by claims 1, 12, and 16.

Claims 15 and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Veditz et al.* ("*Veditz*"), US Patent No. 6,496,793, in view of *Watanabe et al.* ("*Watanabe*"), US Patent No. 6,185,729, and further in view of *Kan et al.*, ("*Kan*"), US PGPUB No. 2003/0088544.

Claims 15 and 25 each depend from one of independent claims 12 or 16. As the discussion above demonstrates that *Veditz*, in view of Watanable, does not disclose the limitations of these independent claims, Applicants submit that dependent claims 15 and 25 are allowable without the need for further comment.

Therefore, the claims are believed to be allowable, and allowance of the claims is respectfully requested.

Conclusion

Having addressed all issues set out in the office action, Applicants respectfully submit that the claims are in condition for allowance and respectfully request that the claims be allowed.

Respectfully submitted, and
S-signed pursuant to 37 CFR 1.4,

/Gero G. McClellan, Reg. No. 44,227/

Gero G. McClellan
Registration No. 44,227
PATTERSON & SHERIDAN, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Applicant(s)